The Da Vinci Code of Large Pre-trained Language Models: Deciphering Degenerate Knowledge Neurons

Anonymous ACL submission

Abstract

This study explores the mechanism of factual knowledge storage in pre-trained language models (PLMs). Previous research suggests that factual knowledge is stored within multi-layer perceptron weights, and some storage units exhibit degeneracy, referred to as Degenerate Knowledge Neurons (DKNs). This paper provides a comprehensive definition of DKNs that covers both structural and functional aspects, pioneering the study of structures in PLMs' factual knowledge storage units. Based on this, we introduce the *Neurological Topology Clustering* method, which allows the formation of DKNs in any numbers and structures, leading to a more accurate DKN acquisition. Furthermore, we introduce the *Neuro-Degeneracy Analytic Analysis Framework*, which uniquely integrates model robustness, evolvability, and complexity for a holistic assessment of PLMs. Within this framework, our execution of 34 experiments across 2 PLMs, 4 datasets, and 6 settings highlights the critical role of DKNs. The code will be available soon.

1 Introduction

Recent studies reveal that large pretrained language models (PLMs) stores extensive factual knowledge (Touvron et al., 2023; OpenAI et al., 2023), yet the mechanisms of knowledge storage in PLMs remain largely unexplored. Dai et al.(2022) propose that some multi-layer perceptron (MLP) modules can store "knowledge". As shown in the Part A of Figure 1, for the fact ⟨ *COVID-19*, *dominant variant*, *Delta* ⟩, the corresponding knowledge storage units a through f are termed knowledge neurons (KNs) (Dai et al., 2022). Chen et al.(2024) find that distinct KN pairs, such as $\{a, b\}$ and $\{c, d\}$ in Part B1 of Figure 1, can store identical facts. They define the set of these pairs as degenerate knowledge neurons (DKNs) from a functional perspective.

While Chen et al.(2024) have conducted some exploration on DKNs, their acquisition method still faces two issues. (1) *Numerical Limitation*: Their method constrains each DKN's element to contain just two KNs, such as $\{a, b\}$ in Part B1 of Figure 1. However, factual knowledge can be stored across more than two neurons (Allen-Zhu and Li, 2023). (2) *Connectivity Oversight*: The connection of DKNs can be either tight or loose, and knowledge within PLMs can be stored either centrally or dispersedly, which affects the expression of knowledge (Zhu and Li, 2023). However, this has been overlooked in prior research of Chen et al.(2024).

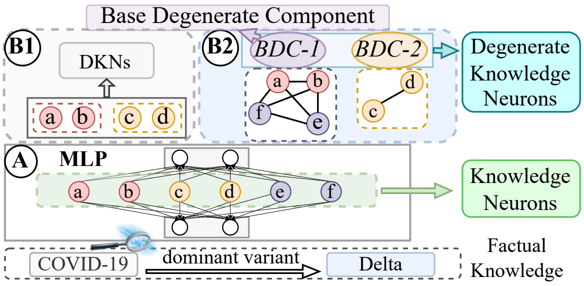

Figure 1: Explanation of KNs and DKNs. Part A represents KNs in the multi-layer perceptron (MLP), while Part B1 and Part B2 symbolize the preliminary and our comprehensive definitions of DKNs.

To address these two issues, we first provide a comprehensive definition of DKNs. **Functionally**, certain subsets of KNs can independently express the same fact, termed as Base Degenerate Components (BDCs), such as *BDC-1* and *BDC-2* in Part B2 of Figure 1. In other words, they exhibit mutual degeneracy. The set of these BDCs is referred to as a DKN. **Structurally**, BDCs like *BDC-1* and *BDC-2* differ in KN number and connection tightness. To assess DKNs' structural traits, we define neuron distances through connection weights and analyze them with an adjacency matrix.

Based on the above definition, we introduce the **Neurological Topology Clustering (NTC)** method, including clustering and filtering stages. This method enables the formation of BDCs with

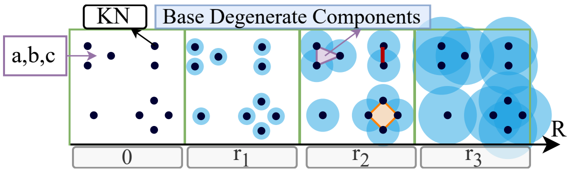

Figure 2: The clustering part of Neurological Topology Clustering method, and x-axis (R) represents the expansion of the radius.

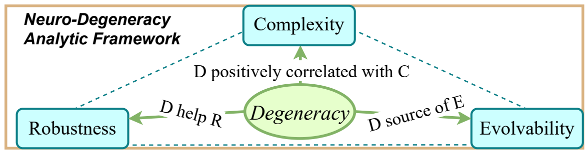

Figure 3: The Neuro-Degeneracy Analytic Framework and the relationship between degeneracy, robustness, evolvability and complexity.

any number of neurons and types of connections. Figure 2 shows its clustering stage. Considering KNs $\{a, b, c\}$, they are isolated points at R=0. From r_1 to r_3, circles centered on KNs expand outward. When R=r_1, the circles do not intersect, and no new clusters are formed. When R=r_2, $\{a, b, c\}$ form a cluster. When R=r_3, all points merge into one cluster, also failing to yield a suitable cluster. During the radius change, a wide range of R values keep $\{a, b, c\}$ clustered together, indicating the set's stable existence. This stable cluster, indicating a strong knowledge expression ability (Zhu and Li, 2023), is identified as a BDC. We then filter BDCs to derive DKNs, detailed in Section 3.

In cognitive science, degeneracy is believed to bridge robustness, evolvability, and complexity (Whitacre and Bender, 2010; Edelman and Gally, 2001; Whitacre, 2010; Mason, 2015). Inspired by this, we propose the **Neuro-Degeneracy Analytic Framework** to study these properties in PLMs through the lens of DKNs, as shown in Figure 3.

(1) **Robustness**: PLMs' robustness relates to their ability to handle errors or input interference (Fernandez et al., 2005). We investigate how DKNs influence PLMs' robustness under two experimental settings. First, we attenuate the values or connection weights of DKNs, and second, we enhance the values or connection weights of DKNs. Experiments demonstrate that **DKNs can help PLMs cope with input interference**. Furthermore, we conduct a fact-checking experiment (Guo et al., 2022), using DKNs to detect false facts.

(2) **Evolvability**: Evolvability is defined as the ability to adaptively evolve in new environments (Kirschner and Gerhart, 1998). Our research explores one aspect of PLMs' evolvability, namely their ability to learn new knowledge. Since factual knowledge is constantly being generated and changed, we hope PLMs can learn new knowledge without forgetting old knowledge. To explore this, we utilize timestamped factual knowledge (Dhingra et al., 2022), and conduct three experiments. First, to prove that PLMs utilize DKNs to learn new knowledge, we directly fine-tune the PLMs and find that the regions of parameter changes highly overlap with DKNs. Second, we consider an efficient fine-tuning method, freezing all parameters except DKNs. We discover that PLMs can utilize DKNs to efficiently learn new knowledge while not forgetting old knowledge. Third, to verify that PLMs have learned genuine knowledge rather than superficial associations, we conduct data augmentation and find that the accuracy of PLMs on the augmented data remains high. Experiments show that **PLMs can efficiently learn new knowledge through DKNs.**

(3) **Complexity**: PLMs' complexity is positively correlated to the number of parameters (Mars, 2022). In the series of experiments mentioned above, we compare the performance of PLMs across different scales and find that **degeneracy is positively correlated with complexity.** Furthermore, we conduct a fact-checking experiment on complex texts, proving that we can combine the large PLMs' complex text understanding ability with the DKNs' fact-checking capability to complete complex task.

Our contributions can be summarized as follows:

- We provide a comprehensive definition of DKNs from both functional and structural aspects, pioneering the study of structures in PLMs' factual knowledge storage units.

- We introduce the Neurological Topology Clustering method, allowing the formation of DKNs in any numbers and structures, leading to a more accurate DKN acquisition.

- We propose the Neuro-Degeneracy Analytic Framework, which uniquely integrates model robustness, evolvability, and complexity for a holistic evaluation of PLMs. Within this framework, we conduct 34 experiments across 2 PLMs and 4 datasets under 6 settings, demonstrating the pivotal role of DKNs.

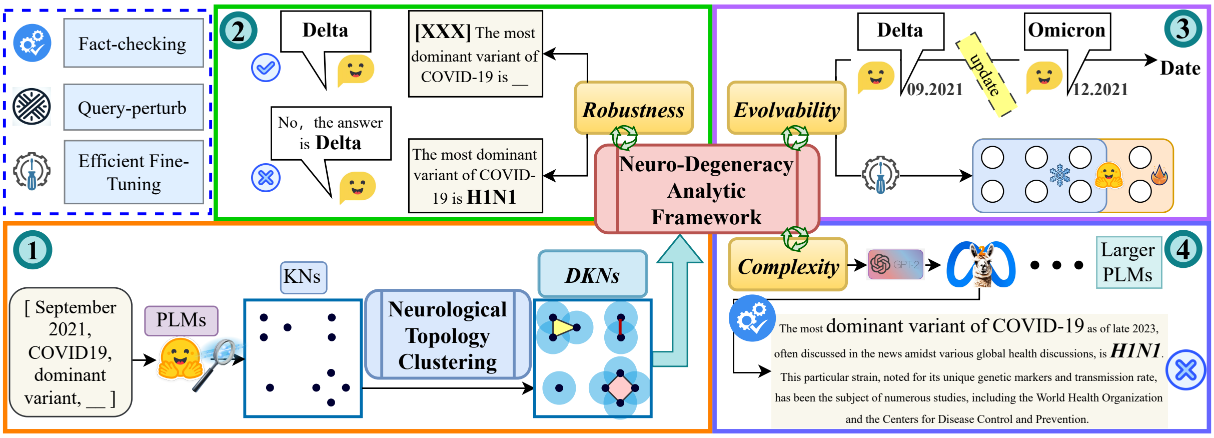

Figure 4: Methodology Overview. *Part 1:* The Neurological Topology Clustering method for DKNs acquisition. *Part 2, 3, 4:* Neuro-Degeneracy Analytic Framework. Within this framework, we conducted a series of experiments to assess the impacts of DKNs on the robustness, evolvability, and complexity of PLMs.

2 Datasets and Task Setups

We utilize the TempLama dataset (Dhingra et al., 2022). Each data instance includes a relation name, a date, a query, and an answer, such as ⟨ *P37*, *September 2021*, *COVID-19*, *dominant variant*, __ ⟩. Except for timestamps, our dataset matches the Lama (Petroni et al., 2019a, 2020) and mLama (Kassner et al., 2021) formats used by Dai et al.(2022) and Chen et al.(2024). We select GPT-2 (Radford et al., 2019) and LLaMA2-7b (Touvron et al., 2023) to test the scalability of our methods, as they share similar architectures but differ in parameter sizes. Our overall method is illustrated in Figure 4. Section 3 introduces the Neurological Topology Clustering method. Under our proposed Neuro-Degeneracy Analytic Framework, Sections 4, 5, and 6 analyze PLMs' robustness, evolvability, and complexity through a series of experiments.

3 Neurological Topology Clustering

3.1 Definition of DKNs

Formalization Considering a fact, we utilize the AMIG method (Chen et al., 2024) to obtain KNs, denoting them as $\mathcal{N} = \{n_1, n_2, \ldots, n_k\}$, where n_i is a KN. For details of this method, see Appendix B. Let DKNs be denoted as $\mathcal{D}$, containing s elements, $\mathcal{D} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_s\}$, where $\mathcal{B}_j = \{n_{j1}, n_{j2}, \ldots, n_{|\mathcal{B}_j|}\}$ is named as the Base Degenerate Component (BDC). Thus, this fact ultimately corresponds to a set of DKNs:

$$\mathcal{D} = \{\mathcal{B}_1, \ldots, \mathcal{B}_s\} = \{(n_{11}, \ldots, n_{|\mathcal{B}_1|}), \ldots, (n_{s1}, \ldots, n_{|\mathcal{B}_s|})\} \tag{1}$$

Functional Definition Degeneracy requires that each BDC should independently express a fact. Let $Prob(\mathcal{B})$ represent the PLMs' predictive probability when $\mathcal{B}$ is activated, then the functional definition of DKNs is:

$$Prob(\mathcal{D}) \approx Prob(\mathcal{B}_i), \forall i = 1, 2, \ldots, s \tag{2}$$

$$Prob(\emptyset) \ll Prob(\mathcal{B}_i), \forall i = 1, 2, \ldots, s \tag{3}$$

Structural Definition Zhu and Li(2023) argue that tightly connected DKNs tend to store knowledge centrally. Thus, we use the adjacency matrix $\mathcal{A}$ to evaluate the DKNs' connection structure. For neurons A and B in layer l_A and layer l_B respectively, we calculate the distance d_{AB} as follows:

$$d_{AB} = \begin{cases} |1/w_{AB}| & \text{if } w_{AB} \neq 0 \text{ and } |l_A - l_B| = 1, \\ \min\limits_{P \in \text{Paths}(\mathcal{N})} \sum\limits_{(i,j) \in P} d_{ij} & \text{if } |l_A - l_B| > 1 \text{ and a path exists,} \\ \infty & \text{otherwise.} \end{cases} \tag{4}$$

where Paths($\mathcal{N}$) includes all paths from A to B through $\mathcal{N}$. Distance calculation varies in three scenarios. First, for neurons in adjacent layers, it is the reciprocal of the weight. Second, for neurons spanning multiple layers, it is the shortest distance determined by a dynamic programming algorithm. Third, for neurons in the same layer, since information in PLMs transmits between layers rather than within a layer (Meng et al., 2022), we set the distance to ∞. Hence, any $\mathcal{D}$ can correspond to an adjacency matrix $\mathcal{A}$, where $\mathcal{A} \in \mathbb{R}^{k \times k}$, and k is the number of knowledge neurons contained in $\mathcal{D}$. Based on $\mathcal{A}$, beyond traditional neuron value editing, modifying the connection weights offers another method to edit PLMs.

3.2 The Acquisition of DKNs

Persistent Homology In our method, we capture the persistent homology (Edelsbrunner et al., 2008)

Method	GPT-2						
	2	3	4	5	6	7	Average
DBSCAN	18 → 39	23 → 34	33 → 53	25 → 50	26 → 47	9 → 13	24.40 → 40.11
Hierarchical	16 → 7.3	3.7 → 12	4.3 → 27	3.3 → 25	25→ 92	—	20.78 → 30.81
K-Means	18 → 30	28 → 45	32 → 59	289 → 31	23 → 30	23 → 34	34.44 → 39.22
AMIG	-0.79 → 0.89	—	—	—	—	—	-0.79 → 0.89
NTC (Ours)	**7.9 → 53**	**8.6 → 44**	**12 → 92**	**11 → 53**	**3.1 → 32**	**7.8 → 61**	**9.32 → 55.60**

Method	LLaMA2						
	2	3	8	11	14	17	Average
DBSCAN	7.1 → 15	8.7 → 15	7.7 → 16	7.2 → 25	—	—	22.26 → 18.24
Hierarchical	28 → 44	22 → 6.5	—	—	—	—	20.8 → 30.8
K-Means	2.8 → 16	4.3 → 19	7.8 → 50	4.6 → 26	14 → 38	31 → 135	34.4 → 39.2
AMIG	-0.99 → 1.99	—	—	—	—	—	-0.99 → 1.99
NTC (Ours)	**2.8 → 16**	**4.3 → 19**	**7.8 → 50**	**4.6 → 26**	**14 → 38**	**31 → 135**	**14.11 → 28.78**

Table 1: Comparison of $\mathcal{D}$ obtained by different methods. The figures atop indicate the number of BDCs. Each cell shows the $\Delta Prob$: left of the arrow for partial BDC attenuation, right for full BDC attenuation. For example, under 5 BDCs, left figure indicates average $\Delta Prob$ for attenuating 1 to 4 BDCs, right figure for all 5 BDCs. The symbol "—" denotes that a specific method failed to yield $\mathcal{D}$ of the specified length, and "Average" is the average results.

Algorithm 1 Neurological Topology Clustering

Require: Knowledge neurons $\mathcal{N}$, Adjacent matrix $\mathcal{A}$, dynamic threshold τ_1 and threshold τ_2
Ensure: Degenerate knowledge neurons $\mathcal{D}$

```
Initialize D = ∅.
R is the radius for persistent homology. Initialize R ← 0.
while R increases do
    Record all potential base degenerate components B_i
    and their corresponding persistence duration R_p.
end while
for each B_i do
    if R_p(B_i) > τ1 and Prob(B_i) > τ2 then
        Add B_i to D, where Prob(B_i) is the predictive
        probability when B_i is activated.
    end if
end for
return D
```

of KN sets, which represents the duration of the set's existence and the tightness of the set's connections. Consider two KNs, n_i and n_j, which are both the centers of expanding circles. When the start radius is 0, this corresponds to $R_s = 0$. Suppose they touch at radius $R_e = r_1$, which marks the end radius $R = r_1$, and a new start radius $R_s = r_1$. At this point, n_i and n_j are clustered together, forming a BDC, $\mathcal{B} = \{n_i, n_j\}$, corresponding to a persistence duration $R_p = R_e - R_s = r_1$. For details on persistent homology, see Appendix C.

NTC Method Our method is shown in Part 1 of Figure 4 and Algorithm 1. We designed two steps, clustering and filtering, to obtain DKNs. During the clustering process, as R increases from 0 to infinity, we record all BDCs along with their corresponding R_p. During the filtering process, we initially select BDCs with R_p above τ_1. Then, among these, only BDCs with a $Prob(\mathcal{B}_i)$ greater than a threshold τ_2 are kept. Finally, these BDCs constitute $\mathcal{D}$, as shown in the formula below.

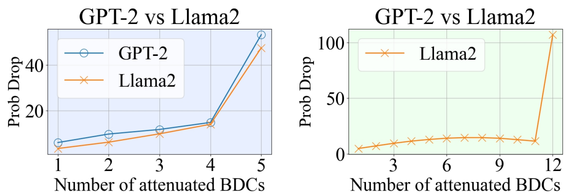

Figure 5: The graph shows $\Delta Prob$ against the number of attenuated BDCs. It is worth noting that the final point represents attenuate all BDCs.

$$\mathcal{D} = \{\mathcal{B}_i | R_p(\mathcal{B}_i) > \tau_1 \text{ and } Prob(\mathcal{B}_i) \geq \tau_2\} \quad (5)$$

3.3 Experiments of DKNs Acquisition

Experimental settings Considering a set $\mathcal{D} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n\}$, we traverse all subsets of $\mathcal{D}$, attenuating values or connection weights of all BDCs in this subset. Then, we calculate the predictive probabilities of the PLMs. Prediction probability change is $\Delta Prob = \frac{Prob_1 - Prob_2}{Prob_1}$, where $Prob_1$ and $Prob_2$ are probabilities before and after attenuation. We selected four other methods as baselines, including K-Means (Ahmed et al., 2020), DBSCAN (Ester et al., 1996), and Hierarchical Clustering (Murtagh and Contreras, 2012) and AMIG (Chen et al., 2024).

Explanation of Figures and Tables Figure 5 shows the variation of $\Delta Prob$ relative to the number of attenuated BDCs. We select cases where the length of $\mathcal{D}$ is 5 and 12. Since GPT-2 does not have $\mathcal{D}$ containing 12 BDCs, it is not depicted. Table 1 presents our overall results. Given the diversity in the number of BDCs contained within $\mathcal{D}$ across different models and methods, we discuss them categorically, presenting only some representative results and the average results. The complete results are reported in the Appendix D.

Result Analysis (A) DKNs identified by NTC exhibit strong degeneracy. The "Average" result in Table 1 and the clear inflection point in Figure 5 suggest our approach has a lower $\Delta Prob$ for attenuating partial BDCs and higher for all. This aligns with KN definitions in Equations 2 and 3. (B) Persistent homology can identify DKNs with better degenerate properties. Our method's DKNs, as evidenced in Table 1, adhere best to the theoretical definitions provided by Equations 2 and 3. (C) Attenuating DKNs could inadvertently boost PLMs' ability to express knowledge. Negative values in Table 1 represents an increased predictive probability. This suggests that attenuating some subsets of DKNs allows others to compensate and improve PLMs' performance under certain conditions.

4 The Impact of DKNs on Robustness

4.1 Query-Perturbation

Explanation In practical scenarios, PLMs often encounter user input errors like spelling mistakes or character omissions (Chen et al., 2010). To simulate this scenario, we apply random disturbances to the inputs. Given an input sequence $Q = \{q_1, q_2, \ldots, q_n\}$, we generate its perturbed counterpart Q^*:

$$Q^* = \begin{cases} \{q_1, \ldots, q_{i-1}, \text{[replace]}, q_{i+1}, \ldots, q_n\} & \text{if replace,} \\ \{q_1, \ldots, q_{i-1}, \text{[add]}, q_i, \ldots, q_n\} & \text{if add,} \\ \{q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_n\} & \text{if delete.} \end{cases} \tag{6}$$

where "[replace]" and "[add]" are special characters. Then we present two sets of experimental setups, as shown in Part 2 of Figure 4.

Attenuating DKNs We apply two different operations to attenuate DKNs. (1) Attenuate Values: zeroing neuron values. (2) Attenuate Weights: nullifying neuron connection weights. Then, we denote the predictive probabilities of PLMs as $Prob$, and calculate $Prob$ for both Q and Q^*, along with their relative decrease. We find that PLMs exhibit diminished robustness when their degeneracy is reduced. As depicted in Figure 6, attenuating DKNs

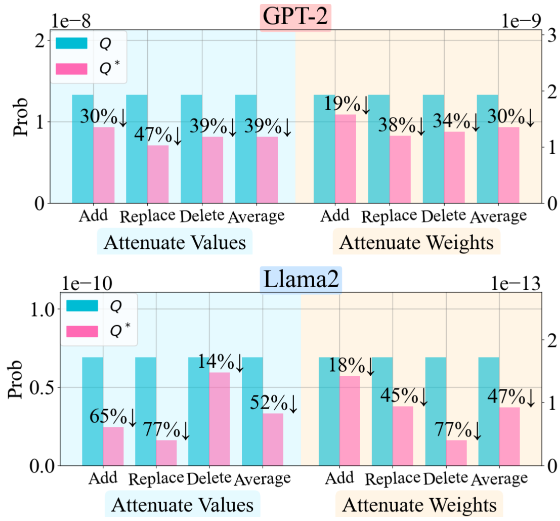

Figure 6: Results of attenuating DKNs experiments. Q and Q^* are the query and the perturbed query. The symbol "↓" represents the reduction in predictive probability, calculated by the formula $\frac{Prob(Q^*)-Prob(Q)}{Prob(Q)}$.

results in a higher $Prob(Q)$ and a lower $Prob(Q^*)$, which indicates PLMs still possess specific knowledge but lack the ability to resist input interference.

Enhancing DKNs Since robustness in PLMs is always limited, they may answer some Q^* queries incorrectly. We record these queries as Q^*_{err} and apply two operations to enhance DKNs. (1) Enhance Values: doubling the value of neurons. (2) Enhance Weights: doubling neuron connection weights. Then, we reassess the enhanced PLMs' accuracy on Q^*_{err}, denoted as Acc_{err}. Differing from the previous metric, although $Prob$ is suitable for evaluating DKNs' effect across all queries, changes in $Prob$ may not affect PLMs' outputs. Thus, to measure enhanced DKNs' impact on PLM performance, a significant change in $Prob$ is necessary, thus leading to the choice of Acc_{err} here.

We find that enhancing DKNs improves the robustness of PLMs. Since the dataset for this experiment is Q^*_{err}, the baseline accuracy can be considered as 0. Table 2 demonstrates that PLMs successfully respond to queries they previously answered incorrectly after enhancement. From the results of the above experiments, we can conclude that **DKNs can help PLMs cope with input interference**.

4.2 Fact-Checking

Explanation Since factual errors can also be seen as a form of perturbation, we naturally think of conducting fact-checking experiments based on DKNs. Unlike knowledge localization where the true value

Method	GPT-2				LLaMA2			
	Add	**Delete**	**Replace**	**Average**	**Add**	**Delete**	**Replace**	**Average**
Enhance Values	0.102	0.118	0.020	0.074	0.185	0.191	0.134	0.168
Enhance Weights	0.120	0.147	0.102	0.120	0.196	0.188	0.131	0.168

Table 2: Results of enhancing DKNs experiments. The table values is accuracy after enhancement (Acc_{err}).

Method	GPT-2 (Relation)			LLaMA2 (Relation)			GPT-2 (Golden)			LLaMA2 (Golden)		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
KNs	0.489	0.511	0.500	0.481	0.455	0.468	0.600	0.636	0.618	0.568	0.741	0.643
PLMs	0.015	0.015	0.015	0.035	0.036	0.036	0.015	0.015	0.015	0.035	0.036	0.036
DKNs	**0.497**	**0.520**	**0.508**	**0.505**	**0.500**	**0.502**	**0.549**	**0.848**	**0.667**	**0.530**	**0.955**	**0.682**

Table 3: Results of the fact-checking experiment. "Relation" represents DKNs or KNs based on relation, and "Golden" represents DKNs or KNs obtained using true answers.

y^* is known, fact-checking does not allow us to know y^* in advance. To address this, we divide the factual knowledge by relation. For queries Q^r corresponding to a relation, we split them into two parts: Q^{r1} for DKNs acquisition and Q^{r2} for testing. For a given query Q_i^{r1}, the corresponding set of DKNs is denoted as D_i^{r1}. We aggregate them and select those that appear more than τ_3 times, denoting as the relation-based DKNs D^r:

$$D^r = \{n_i \mid n_i \in \bigcup_{D_i^{r1}} \text{ and Count}(n_i) > \tau_3\} \tag{7}$$

where Count(n_i) is the number of occurrences of n_i. Then, for a query Q_i^{r2}, we determine the factual correctness by computing the average attribution score of all neurons in $\mathcal{D}^r$:

$$FC(Q^{r2}) = \begin{cases} \text{True} & \text{if} \sum_{n \in \mathcal{D}^r} Score(n) \,/\, |\mathcal{D}^r| > \tau_4, \\ \text{False} & \text{otherwise.} \end{cases} \tag{8}$$

where $Score(n)$ denotes the activation score of each neuron n in $\mathcal{D}^r$, τ_4 is a threshold.

Experimental settings We first construct a dataset Q_f^{r2} by substituting correct answers within Q_i^{r2} with an alternate answer from Q_j^{r2}, and perform fact-checking on Q_f^{r2}. Then, we employ two baseline methods: (1) Fact-checking based on KNs, and (2) Direct fact-checking with PLMs providing True or False answers. Besides using relation-based $\mathcal{D}^r$, we also utilize $\mathcal{D}_i^r$ obtained from true values. For evaluation metrics, we choose Precision (P), Recall (R), and F1-Score.

Results Analysis (A) DKNs have the strongest fact-checking ability, as indicated by the results in Table 3. (B) The more precise the location of DKNs, the better the fact-checking performance. This is evidenced by the "Golden" results in Table 3, which outperform the "Relation" results. (C) The fact-checking ability of PLMs is weak. PLMs may correctly answer a query but fail to identify its errors, likely because of insufficient familiarity with certain facts. Since true values are used in obtaining D^r, the fact-checking ability for such facts is stronger.

5 The Impact of DKNs on Evolvability

As Part 3 of 4 shows, degeneracy is key to model evolvability, enabling adaptation to new environments while preserving existing functions. We investigates one aspect of evolvability, namely the ability of PLMs to learn new knowledge. For example, given a timestamped query, *"In date __, the dominant variant of COVID-19 is __"*, a PLM may update its answer from "Delta" to "Omicron" across different timestamps, while still retaining the knowledge of the "Delta" variant.

5.1 Overlap of DKNs and Parameter Changes

Experimental settings We first directly fine-tuning the PLMs and then record the positions of neurons where significant parameter changes occur, denoted as ΔN:

$$\Delta N = \{n \,|\, \Delta P(n) > \tau_{\Delta N}\} \tag{9}$$

$$\Delta P(n) = \sqrt{\left(\frac{\|w_2^{\text{fc}}(n) - w_1^{\text{fc}}(n)\|}{\|w_1^{\text{fc}}(n)\|}\right)^2 + \left(\frac{\|w_2^{\text{proj}}(n) - w_1^{\text{proj}}(n)\|}{\|w_1^{\text{proj}}(n)\|}\right)^2} \tag{10}$$

where $\tau_{\Delta N}$ is a dynamic threshold, $\Delta P(n)$ indicates parameter change. $w_1^{\text{fc}}(n)$ and $w_1^{\text{proj}}(n)$ signify the feed-forward and projection weights of neuron n before fine-tuning, respectively, while

Method	GPT-2				LLaMA2			
	Q_{new}	Q_{old}	Q_{au}	Average	Q_{new}	Q_{old}	Q_{au}	Average
$\Theta(\mathcal{N})$	0.85	0.86	0.82	0.84	0.95	0.91	0.92	0.93
$\Theta(Rnd)$	0.43	0.71	0.38	0.51	0.78	0.88	0.71	0.79
$\Theta(All)$	0.93	0.49	0.91	0.78	0.99	0.66	0.99	0.89
$\boldsymbol{\Theta}(\mathcal{D})$	0.88	0.83	0.86	**0.86**	0.98	0.93	0.97	**0.96**

Table 4: Results of the fine-tuning experiment. Θ denotes the areas unfrozen in PLMs during fine-tuning, while Q_{new}, Q_{old}, and Q_{au} symbolize new, old, and augmented queries, respectively.

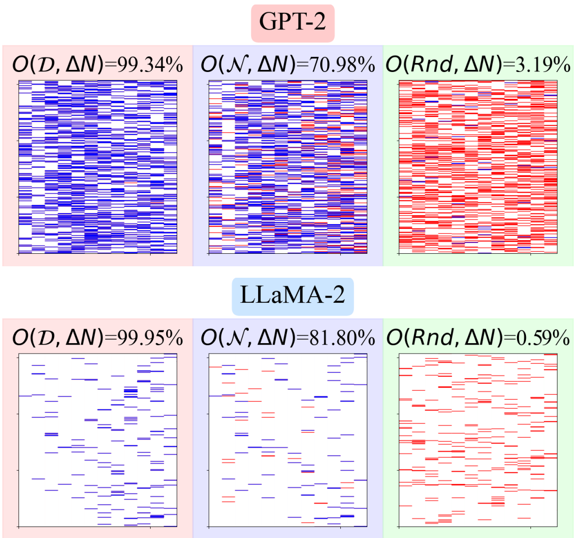

Figure 7: Results of the parameter changes experiment. Blue represents the overlap between neurons and the area of parameter changes, while red indicates the non-overlapping portions. The degree of overlap O is also reported in the figure.

$w_2^{\text{fc}}(n)$ and $w_2^{\text{proj}}(n)$ are their post-fine-tuning counterparts. Then, we identify the corresponding $\mathcal{D}$ through Algorithm 1, and calculate the overlap between $\mathcal{D}$ and ΔN:

$$O(\mathcal{D}, \Delta N) = |\mathcal{D} \cap \Delta N| \,/\, |\mathcal{D}| \tag{11}$$

For comparison, we choose the KNs ($\mathcal{N}$) and randomly chosen neurons (Rnd) as baselines.

Results Analysis The PLMs indeed utilize DKNs to learn new knowledge. Figure 7 reveals $O(\mathcal{D}, \Delta N)$ is highest, indicating that the parameter change area has the largest overlap degree with DKNs. Notably, LLaMA2 has more irrelevant neurons due to its larger number of parameters, but this does not conflict with its high $O(\mathcal{D}, \Delta N)$.

5.2 DKNs Guide PLMs to Learn Knowledge

Experimental settings Given a dataset and corresponding $\mathcal{D}$, we freeze the parameters outside of $\mathcal{D}$ during fine-tuning. Then we evaluate the role of DKNs in learning new knowledge by conducting the experiments with three distinct datasets. (1) Q_{new}: Comprising queries introducing new knowledge, aimed at evaluating PLMs' grasp of it. (2) Q_{old}: Comprising knowledge previously mastered by PLMs, with no overlap with Q_{new}, used to assess if PLMs have forgotten old knowledge. (3) Q_{au}: Comprising augmented data, which rephrases Q_{new} into semantically identical but differently expressed queries. We use it to prove that PLMs actually learn knowledge, not just superficial semantic association. Besides unfreezing DKNs, denoted as $\Theta(\mathcal{D})$, we also select three baseline methods. (1) Unfreezing KNs, denoted as $\Theta(\mathcal{N})$. (2) Unfreezing a random set of neurons equal in number to the DKNs, denoted as $\Theta(Rnd)$. (3) Unfreezing all neurons, i.e., direct fine-tuning, denoted as $\Theta(All)$.

Results Analysis (A) **DKNs can guide PLMs to learn new knowledge more effectively while preserving old knowledge**. Table 4 shows while direct fine-tuning can yield high accuracy on Q_{new}, it may cause a markedly reduced accuracy on Q_{old}. However, unfreezing DKNs not only maintains high accuracy on Q_{new} but also significantly improves performance on Q_{old} in comparison to direct fine-tuning. This approach offers a new potential solution to the problem of catastrophic forgetting and makes the fine-tuning process more efficient. (B) Compared to other neurons, DKNs possess the strongest ability to guide the PLMs in learning new knowledge. This conclusion is drawn from comparing the accuracy of different methods in Table 4. (C) PLMs can learn genuine knowledge rather than just superficial semantic information. This assertion is supported by the close accuracy rates between Q_{au} and Q_{new}.

6 The Impact of DKNs on Complexity

6.1 Explanation of Complexity

Hypotheses In auto-regressive models, complexity, denoted by $\mathcal{C}(M)$, is determined by the number of model parameters (Hu et al., 2021). Consider two PLMs, M_1 and M_2. Let $\mathcal{D}^*(M)$ be the PLMs' degeneracy, $\mathcal{T}$ be a downstream task using DKNs. The performance of M on $\mathcal{T}$ is indicated by $\mathcal{P}(M, \mathcal{T})$, and $Param(M)$ is the number of parameters in M. We propose two hypotheses:

$$\begin{aligned} &\textbf{Hypothesis 1} \quad Param(M_1) < Param(M_2) \Rightarrow \mathcal{C}(M_1) < \mathcal{C}(M_2) \\ &\textbf{Hypothesis 2} \quad \mathcal{P}(M_1, \mathcal{T}) < \mathcal{P}(M_2, \mathcal{T}) \Rightarrow \mathcal{D}^*(M_1) < \mathcal{D}^*(M_2) \end{aligned} \tag{12}$$

Results Analysis The overarching conclusion is that **degeneracy is positively correlated with complexity**. By comparing the performances of PLMs in the aforementioned experiments, we observe the following phenomena. (A) When DKNs are attenuated, LLaMA2 experiences a more significant drop in predictive probability compared to GPT-2. Conversely, enhancing DKNs leads to a more notable increase in LLaMA2's accuracy, as illustrated in Figure 6 and Table 2. (B) LLaMA2 surpasses GPT-2 in fact-checking abilities when utilizing DKNs, as indicated by Table 3. (C) Compared to GPT-2, LLaMA2 has a stronger ability to learn new knowledge, retain old knowledge, and acquire genuine knowledge, as detailed in Table 4. The aforementioned phenomena indicate that $\mathcal{P}(\text{GPT-2}, \mathcal{T}) < \mathcal{P}(\text{LLaMA2}, \mathcal{T})$, and given that $Param(\text{GPT-2}) < Param(\text{LLaMA2})$, according to Hypotheses 12, we can conclude:

$$\mathcal{C}(\text{GPT-2}) < \mathcal{C}(\text{LLaMA2}) \Rightarrow \mathcal{D}^*(\text{GPT-2}) < \mathcal{D}^*(\text{LLaMA2}) \tag{13}$$

6.2 Fact-Checking within Complex Texts

Experimental settings We investigate the potential of DKNs when integrated with PLMs' other abilities, through an experiment that melds the model's ability to comprehend complex texts with DKNs' fact-checking ability within such texts. Consider a triplet-form query Q, first, we rewrite it into a complex text Q^c. To perform fact-checking on Q^c, the PLMs must possess both reasoning ability and fact-checking ability. Then, we provide a prompt to PLMs, requiring them to first identify both the query and answer in Q^c, and then perform fact-checking according to Equation 8.

Results Analysis (A) PLMs, with the ability to understand complex texts, can employ DKNs for fact-checking within complex texts. Table 5 shows LLaMA2's ability to utilize DKNs for fact-checking. However, GPT-2 only echoes prompts without yielding meaningful results, thus its exclusion from the table. (B) When DKNs' locations are imprecise, pre-understanding complex texts can enhance fact-checking ability. The results of "Relation" in Table 5 surpass those in Table 3 (F1: 0.56 to 0.502), but the results of "Golden" decrease (F1: 0.62 to 0.682). This indicates that relation-based DKNs lack precision, but complex text understanding increases PLMs' sensitivity to specific facts.

Method	LLaMA2(Relation) P	LLaMA2(Relation) R	LLaMA2(Relation) F1	LLaMA2 (Golden) P	LLaMA2 (Golden) R	LLaMA2 (Golden) F1
KNs	0.59	0.48	0.53	0.64	0.43	0.51
PLMs	0.02	0.02	0.02	0.02	0.02	0.02
DKNs	**0.54**	**0.58**	**0.56**	**0.52**	**0.78**	**0.62**

Table 5: Results of the fact-checking in complex texts.

7 Related Work

Petroni et al.(2019b) argue that numerous factual knowledge exists within PLMs and suggest using "fill-in-the-blank" cloze tasks determine if the models have grasped specific facts. Meanwhile, Geva et al.(2021) suggest that MLP modules within transformer models function akin to key-value memory systems. Building on this, Dai et al.(2022) employ the "fill-in-the-blank" cloze tasks and uncover that some MLP module keys and values are capable of storing factual knowledge, termed as *knowledge neurons* (KNs). Lundstrom et al.(2022) confirm the reliability of their knowledge localization method, while subsequent knowledge editing experiments by Meng et al.(2022) and Meng et al.(2023) reinforce that MLP modules indeed store factual knowledge. Building on these, Geva et al.(2023) delve into the operational dynamics of KNs, and Chen et al.(2024) discover that multiple distinct sets of KNs can store identical facts and term these sets as degenerate knowledge neurons (DKNs).

8 Conclusion

This study delves into the mechanisms of factual knowledge storage in PLMs. First, We provide a comprehensive definition of DKNs that covers both structural and functional aspects, pioneering the study of the internal structures of PLMs. Based on this, we introduce the neurological topology clustering method for more precise DKN acquisition. Finally, we propose the neuro-degeneracy analytic framework, conduct extensive experiments, and thoroughly examine model robustness, evolvability, and complexity. Our research demonstrates the critical role of DKNs in PLMs.

Limitations

First, limited by computational resources, our study involves only the GPT-2 and LLaMA2-7b models. To further validate the scalability of our methods and conclusions, it is imperative to conduct studies on larger models, such as LLaMA2-13b and LLaMA2-70b. Second, our research is confined to factual knowledge, and whether similar findings apply to other types of knowledge remains to be investigated. Finally, the generalizability of our findings across different languages and cultural contexts remains an open question. Our study utilizes datasets in English, limiting our ability to assess the performance and applicability of our methods across non-English languages and datasets that encompass a broader range of cultural knowledge.

Ethics Statement

Our research aims to deepen the understanding and enhance the functionality of PLMs by investigating the role of DKNs. While our research seeks to enhance the understanding and functionality of PLMs by exploring the role of DKNs, we are acutely aware of the potential for misuse of these findings. The increased capabilities of PLMs, driven by insights into DKNs, could potentially be exploited for generating misleading information, manipulating public opinion, or other malicious purposes. It is not our intention to facilitate such activities. Instead, our goal is to contribute to the scientific community's knowledge base, enabling the development of more robust, accurate, and ethically aligned language technologies..

We emphasize the responsibility of using these insights ethically, to avoid the exploitation of enhanced PLM capabilities for harmful purposes. To this end, we advocate for transparency in research, collaboration across disciplines for ethical oversight, and the development of regulatory frameworks to ensure that advancements in PLM technology contribute positively to society.

References

Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. 2020. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, 9(8):1295.

Zeyuan Allen-Zhu and Yuanzhi Li. 2023. Physics of language models: Part 3.2, knowledge manipulation. *ArXiv preprint*, abs/2309.14402.

Serguei Barannikov. 1994. The framed morse complex and its invariants. *Advances in Soviet Mathematics*, 21:93–116.

Gunnar Carlsson. 2009. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308.

Tianyi Chen, Yeliz Yesilada, and Simon Harper. 2010. What input errors do you experience? typing and pointing errors of mobile web users. *International journal of human-computer studies*, 68(3):138–157.

Yuheng Chen, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2024. Journey to the center of the knowledge neurons: Discoveries of language-independent knowledge neurons and degenerate knowledge neurons. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. 2005. Stability of persistence diagrams. In *Proceedings of the twenty-first annual symposium on Computational geometry*, pages 263–271.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

Tamal K Dey, Dayu Shi, and Yusu Wang. 2019. Simba: An efficient tool for approximating rips-filtration persistence via sim plicial ba tch collapse. *Journal of Experimental Algorithmics (JEA)*, 24:1–16.

Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. 2022. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273.

Gerald M Edelman and Joseph A Gally. 2001. Degeneracy and complexity in biological systems. *Proceedings of the National Academy of Sciences*, 98(24):13763–13768.

Herbert Edelsbrunner, John Harer, et al. 2008. Persistent homology-a survey. *Contemporary mathematics*, 453(26):257–282.

Herbert Edelsbrunner and John L Harer. 2022. *Computational topology: an introduction*. American Mathematical Society.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.

Jean-Claude Fernandez, Laurent Mounier, and Cyril Pachon. 2005. A model-based approach for robustness testing. In *Testing of Communicating Systems: 17th IFIP TC6/WG 6.1 International Conference, TestCom 2005, Montreal, Canada, May 31-June, 2005. Proceedings 17*, pages 333–348. Springer.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. 2022. A survey on automated fact-checking. *Transactions of the Association for Computational Linguistics*, 10:178–206.

Xia Hu, Lingyang Chu, Jian Pei, Weiqing Liu, and Jiang Bian. 2021. Model complexity of deep learning: A survey.

Nora Kassner, Philipp Dufter, and Hinrich Schütze. 2021. Multilingual LAMA: Investigating knowledge in multilingual pretrained language models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3250–3258, Online. Association for Computational Linguistics.

Michael Kerber and Raghvendra Sharathkumar. 2013. Approximate čech complex in low and high dimensions. In *International Symposium on Algorithms and Computation*, pages 666–676. Springer.

Marc Kirschner and John Gerhart. 1998. Evolvability. *Proceedings of the National Academy of Sciences*, 95(15):8420–8427.

Daniel D Lundstrom, Tianjian Huang, and Meisam Razaviyayn. 2022. A rigorous study of integrated gradients method and extensions to internal neuron attributions. In *International Conference on Machine Learning*, pages 14485–14508. PMLR.

Mourad Mars. 2022. From word embeddings to pre-trained language models: A state-of-the-art walkthrough. *Applied Sciences*, 12(17):8805.

Paul H Mason. 2015. Degeneracy: Demystifying and destigmatizing a core concept in systems biology. *Complexity*, 20(3):12–21.

Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*.

Fionn Murtagh and Pedro Contreras. 2012. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97.

OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2023. Gpt-4 technical report.

Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington. 2017. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6:1–38.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. In *Automated Knowledge Base Construction*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019a. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019b. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Alessandro Verri, Claudio Uras, Patrizio Frosini, and Massimo Ferri. 1993. On the use of size functions for shape analysis. *Biological cybernetics*, 70(2):99–107.

James Whitacre and Axel Bender. 2010. Degeneracy: a design principle for achieving robustness and evolvability. *Journal of theoretical biology*, 263(1):143–153.

James M Whitacre. 2010. Degeneracy: a link between evolvability, robustness and complexity in biological systems. *Theoretical Biology and Medical Modelling*, 7:1–17.

Zeyuan Allen Zhu and Yuanzhi Li. 2023. Physics of language models: Part 3.1, knowledge storage and extraction. *ArXiv preprint*, abs/2309.14316.

Afra Zomorodian and Gunnar Carlsson. 2004. Computing persistent homology. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 347–356.

A Experimental Hyperparameters

A.1 Hardware spcification and environment.

We ran our experiments on the machine equipped with the following specifications:

- CPU: Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz, Total CPUs: 56
- GPU: NVIDIA GeForce RTX 3090, 24576 MiB (10 units)
- Software:
  - Python Version: 3.10.10
  - PyTorch Version: 2.0.0+cu117

Given a complex statement, simplify it into a straightforward sentence that clearly states the subject, predicate, and object. The simplified sentence should capture the key information from the complex statement in a concise format. Here are a few examples to illustrate the conversion:

1. Complex: _X_, renowned for their career in sports, plays for a team or club. This association is a significant part of their professional journey, reflecting their skill and dedication in their field.
   Simplified: Valentino Rossi plays for _X_.
2. Complex: _X_, a key figure in their domain, holds an important position. This role is marked by responsibilities and influence, shaping the direction and success of their organization or country.
   Simplified: Robert Mugabe holds the position of _X_.
3. Complex: _X_, known for their intellectual and professional achievements, attended a notable institution. Their time at this institution played a crucial role in shaping their career and perspectives.
   Simplified: Malala Yousafzai attended _X_.
4. Complex: As the chair of _X_, this individual is a prominent figure in the business world. This position highlights their leadership and strategic vision in guiding the company towards growth and innovation.
   Simplified: _X_ is the chair of bp.
5. Complex: _X_, a figure of significant influence and ideals, is a member of a notable group or organization. Their membership reflects their commitment to certain values and goals, contributing to the group's impact.
   Simplified: Alexei Navalny is a member of the _X_.
6. Complex: _X_ works for a notable company or organization, playing a pivotal role in driving innovation and excellence in their field.
   Simplified: Elon Musk works for _X_.
7. Complex: As the head of the government of _X_, this political figure plays a central role in shaping the nation's policies and international relations.
   Simplified: _X_ is the head of the government of France.
8. Complex: The head coach of _X_ is known for their expertise in coaching, with leadership and strategic skills crucial in steering the team to success.
   Simplified: _X_ is the head coach of Philadelphia Eagles.
9. Complex: _X_, a significant player in its sector, is owned by a larger entity. This ownership structure is key to understanding the company's operations and market influence.
   Simplified: Google is owned by _X_.

Now, given the following complex statement, apply the same process to simplify it:

Figure 8: The prompt of fact-checking experiments on complex texts.

A.2 Experimental Hyperparameters of Neurological Topology Clustering

In Section 3, where we obtain degenerate knowledge neurons, the primary hyperparameters are τ_1 and τ_2. First, τ_1 is a dynamic threshold, set as

$$\tau_1 = 0.5 \times \max\left(R_{\text{persist}}(\mathcal{B}_1), \ldots, R_{\text{persist}}(\mathcal{B}_n)\right) \tag{14}$$

Then, τ_2 is a fixed value.

$$\tau_2 = 0.3 \tag{15}$$

In this experiment, some data led to excessively large changes in predictive probability, indicating that the PLMs had not originally mastered this factual knowledge, resulting in a very low initial predictive probability. To study the storage mechanism of factual knowledge, it's essential to investigate facts already grasped by the model. Therefore, we set a threshold to exclude data that caused extreme changes in predictive probability. If the change in predictive probability $\Delta Prob$ satisfies:

$$\Delta Prob > 900 \tag{16}$$

then that data is excluded.

A.3 Experimental Hyperparameters of Degneracy and Robustness

In Section 4, the first experiment under Query-Perturbation, namely the Attenuating DKN experiment, similar to A.2, excludes data that satisfies the condition:

$$Prob_{\text{atte}} > 900 \tag{17}$$

In the Fact-Checking experiment, τ_3 is similar to τ_1 as a dynamic threshold. We first count the total number of neurons, then set

$$\tau_3 = 0.7 \times N_{\text{total}} \tag{18}$$

where N_{total} represents the total number of neurons.

A.4 Experimental Hyperparameters of Degeneracy and Evolvability

In Section 5, the experimental hyperparameter $\tau_{\Delta N}$ for the Overlap of DKN and Parameter Changes experiment is set as a dynamic threshold. The process involves calculating the maximum value of $\Delta P(n)$ according to Equation 10. Once this value is determined, $\tau_{\Delta N}$ is set differently based on the model in use. For the GPT-2 model, the threshold $\tau_{\Delta N}$ is calculated as:

$$\tau_{\Delta N} = 0.04 \times \max(\Delta P(n_1), \Delta P(n_2), \ldots, \Delta P(n_k)) \tag{19}$$

In contrast, for the Llama2 model, the calculation of the threshold $\tau_{\Delta N}$ is slightly adjusted:

$$\tau_{\Delta N} = 0.05 \times \max(\Delta P(n_1), \Delta P(n_2), \ldots, \Delta P(n_k)) \tag{20}$$

This distinction in the calculation of $\tau_{\Delta N}$ reflects the specific characteristics and performance considerations of each model.

A.5 Experimental Hyperparameters of Degneracy and Complexity

In Section 6, during our fact-checking experiments on complex texts in Section 6, we provide PLMs with a prompt requiring them to first extract factual knowledge in the form of triples and then perform fact-checking using DKNs. Our prompt is as follows:

A.6 Fact-Checking of Complex Texts in Section 6

During our fact-checking experiments on complex texts in Section 6, we prompt PLMs to simplify complex statements into straightforward sentences. Our prompt is shown in Figure 8.

B Knowldege Localization

This section introduces the method we use to acquire knowledge neurons. We employ the approach proposed by Chen et al.(2024), which we will detail below.

Given a query q, we can define the probability of the correct answer predicted by a PLMs as follows:

$$\mathrm{F}(\hat{w}_j^{(l)}) = p(y^*|q, w_j^{(l)} = \hat{w}_j^{(l)}) \tag{21}$$

Here, y^* represents the correct answer, $w_j^{(l)}$ denotes the j-th neuron in the l-th layer, and $\hat{w}_j^{(l)}$ is the specific value assigned to $w_j^{(l)}$. To calculate the attribution score for each neuron, we employ the technique of integrated gradients.

To compute the attribution score of a neuron $w_j^{(l)}$, we consider the following formulation:

$$\mathrm{Attr}(w_j^{(l)}) = (\overline{w}_j^{(l)} - {w'}_j^{(l)}) \int_0^1 \frac{\partial\, \mathrm{F}({w'}_j^{(l)} + \alpha(\overline{w}_j^{(l)} - {w'}_j^{(l)}))}{\partial w_j^{(l)}}\, d\alpha \tag{22}$$

Here, $\overline{w}_j^{(l)}$ represents the actual value of $w_j^{(l)}$, ${w'}_j^{(l)}$ serves as the baseline vector for $w_j^{(l)}$. The term $\frac{\partial\, \mathrm{F}({w'}_j^{(l)} + \alpha(w_j^{(l)} - {w'}_j^{(l)}))}{\partial w_j^{(l)}}$ computes the gradient with respect to $w_j^{(l)}$.

Next, we aim to obtain ${w'}_j^{(l)}$. Starting from the sentence q, we acquire a baseline sentence and then encode this sentence as a vector.

Let the baseline sentence corresponding to q_i be q'_i, and q'_i consists of m words, maintaining a length consistent with q, denoted as $q'_i = (q'_{i1} \ldots q'_{ik} \ldots q'_{im})$. Since we are using auto-regressive models, according to Chen et al.(2024)' method, $q'_{ik} = \langle \text{eos} \rangle$, where $\langle \text{eos} \rangle$ represents "end of sequence" in auto-regressive models.

The attribution score $Attr_i(w_j^{(l)})$ for each neuron, given the input q_i, can be determined using Equation (22). For the computation of the integral, the Riemann approximation method is employed:

$$Attr_i(w_j^l) \approx \frac{\overline{w}_j^{(l)}}{N} \sum_{k=1}^{N} \frac{\partial F({w'}_j^{(l)} + \frac{k}{N} \times (\overline{w}_j^{(l)} - {w'}_j^{(l)})}{\partial w_j^{(l)}} \tag{23}$$

where N is the number of approximation steps.

Then, the attribution scores for each word q_i are aggregated and subsequently normalized:

$$Attr(w_j^l) = \frac{\sum_{i=1}^{m} Attr_i(w_j^l)}{\sum_{j=1}^{n} \sum_{i=1}^{m} Attr_i(w_j^l)}, \tag{24}$$

Let $\mathcal{N}$ be the set of neurons classified as knowledge neurons based on their attribution scores exceeding a predetermined threshold τ, for a given input q. This can be formally defined as:

$$\mathcal{N} = \left\{ w_j^{(l)} \,\middle|\, Attr(w_j^{(l)}) > \tau \right\} \tag{25}$$

where l encompassing all layers and j including all neurons within each layer.

C Persistent homology

Persistent homology is a method for computing topological features of a space at different spatial resolutions. More persistent features are detected over a wide range of spatial scales and are deemed more likely to represent true features of the underlying space rather than artifacts of sampling, noise, or particular choice of parameters (Carlsson, 2009).

To find the persistent homology of a space, the space must first be represented as a simplicial complex. A distance function on the underlying space corresponds to a filtration of the simplicial complex, that is a nested sequence of increasing subsets. One common method of doing this is via taking the sublevel filtration of the distance to a point cloud, or equivalently, the offset filtration on the point cloud and taking its nerve in order to get the simplicial filtration known as Čech filtration (Kerber and Sharathkumar, 2013). A similar construction uses a nested sequence of Vietoris–Rips complexes known as the Vietoris–Rips filtration (Dey et al., 2019).

C.1 Definition

In persistent homology, formally, we consider a real-valued function defined on a simplicial complex, denoted as $f : K \to \mathbb{R}$. This function is required to be non-decreasing on increasing sequences of faces, meaning that for any two faces σ and τ in K, if σ is a face of τ, then $f(\sigma) \leq f(\tau)$.

For every real number a, the sublevel set $K_a = f^{-1}((-\infty, a])$ forms a subcomplex of K. The values of f on the simplices in K create an ordering of these sublevel complexes, which leads to a filtration:

$$\emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_n = K \tag{26}$$

Within this filtration, for $0 \leq i \leq j \leq n$, the inclusion $K_i \hookrightarrow K_j$ induces a homomorphism on the simplicial homology groups for each dimension p, noted as $f_p^{i,j} : H_p(K_i) \to H_p(K_j)$. The p^{th} persistent homology groups are the images of these homomorphisms, and the p^{th} persistent Betti numbers $\beta_p^{i,j}$ are defined as the ranks of these groups (Edelsbrunner and Harer, 2022). Persistent Betti numbers for $p = 0$ coincide with the size function, an earlier concept related to persistent homology (Verri et al., 1993).

The concept extends further to any filtered complex over a field F. Such a complex can be transformed into its canonical form, which is a direct sum of filtered complexes of two types: one-dimensional complexes with trivial differential (expressed as $d(e_{t_i}) = 0$) and two-dimensional complexes with trivial homology (expressed as $d(e_{s_j+r_j}) = e_{r_j}$) (Barannikov, 1994).

A persistence module over a partially ordered set P consists of a collection of vector spaces U_t, indexed by P, along with linear maps $u_t^s : U_s \to U_t$ for $s \leq t$. This module can be viewed as a functor from P to the category of vector spaces or R-modules. Persistence modules over a field F indexed by $\mathbb{N}$ can be expressed as:

$$U \simeq \bigoplus_i x^{t_i} \cdot F[x] \oplus \left(\bigoplus_j x^{r_j} \cdot (F[x]/(x^{s_j} \cdot F[x])) \right) \tag{27}$$

Here, multiplication by x represents a forward step in the persistence module. The free parts correspond to homology generators that appear at a certain filtration level and persist indefinitely, whereas torsion parts correspond to those that appear at a filtration level and last for a finite number of steps (Barannikov, 1994; Zomorodian and Carlsson, 2004).

This framework allows the unique representation of the persistent homology of a filtered simplicial complex using either a persistence barcode or a persistence diagram. In the barcode, each persistent generator is represented by a line segment starting and ending at specific filtration levels, while in the diagram, each generator is represented as a point with coordinates indicating its birth and death times. Barannikov's canonical form offers an equivalent representation.

C.2 Stability

The stability of persistent homology is a key attribute, particularly in its application to data analysis, as it ensures robustness against small perturbations or noise in the data (Cohen-Steiner et al., 2005). This stability is quantitatively defined in terms of the **bottleneck distance**, a metric for comparing persistence diagrams.

The bottleneck distance between two persistence diagrams X and Y is defined as:

$$W_\infty(X, Y) := \inf_{\varphi : X \to Y} \sup_{x \in X} \|x - \varphi(x)\|_\infty \tag{28}$$

where the infimum is taken over all bijections φ from X to Y. This metric essentially measures the greatest distance between matched points (or generators) in two persistence diagrams, considering the optimal matching.

A fundamental result in the theory of persistent homology is that small changes in the input data (such as a filtration of a space) result in small changes in the corresponding persistence diagram, as measured by the bottleneck distance. This is formalized by considering a space X, homeomorphic to a simplicial complex, with a filtration determined by the sublevel sets of a continuous tame function $f : X \to \mathbb{R}$. The map D that takes the function f to the persistence diagram of its kth homology is 1-Lipschitz with respect to the supremum norm on functions and the bottleneck distance on persistence diagrams. Formally, this is expressed as (Cohen-Steiner et al., 2005):

$$W_\infty(D(f), D(g)) \leq \|f - g\|_\infty \qquad (29)$$

This Lipschitz condition implies that a small change in the function f, as measured by the supremum norm, will not cause a disproportionately large change in the persistence diagram. Consequently, persistent homology is particularly useful in applications where data may be subject to noise or small variations, as the essential topological features (captured by the persistence diagrams) are not overly sensitive to such perturbations.

C.3 Computation

There are various software packages for computing persistence intervals of a finite filtration (Otter et al., 2017). The principal algorithm is based on the bringing of the filtered complex to its canonical form by upper-triangular matrices (Barannikov, 1994).

D Complete Experimental Results of Neurological Topology Clustering

In Section 3, it is mentioned that the lengths of DKNs vary. For ease of presentation, we have only shown cases with a larger amount of data. Here, we provide the complete table, as shown in Table 6, which is generated under the same settings as used for Table 1.

For our method, we have also created line charts for DKNs of each length, serving as a supplement to Figure 5. We use the same background color as the figures in the main text to represent images that are completely consistent with the main text, while a white background is used for the line charts corresponding to DKNs of other lengths. Figure 9 show the results corresponding to DKN obtained using the NTC method. Additionally, we also present the results corresponding to other methods. Figure 10 display the results corresponding to DKN obtained using the DBSCAN method. Figure 11 show the results corresponding to DKN obtained using the Hierarchical method. Figures 12 and 13 illustrate the results corresponding to DKN obtained using the K-Means method. Since the DKN length for the method corresponding to AMIG (Chen et al., 2024) is limited to 2, there is no need to display the inflection points.

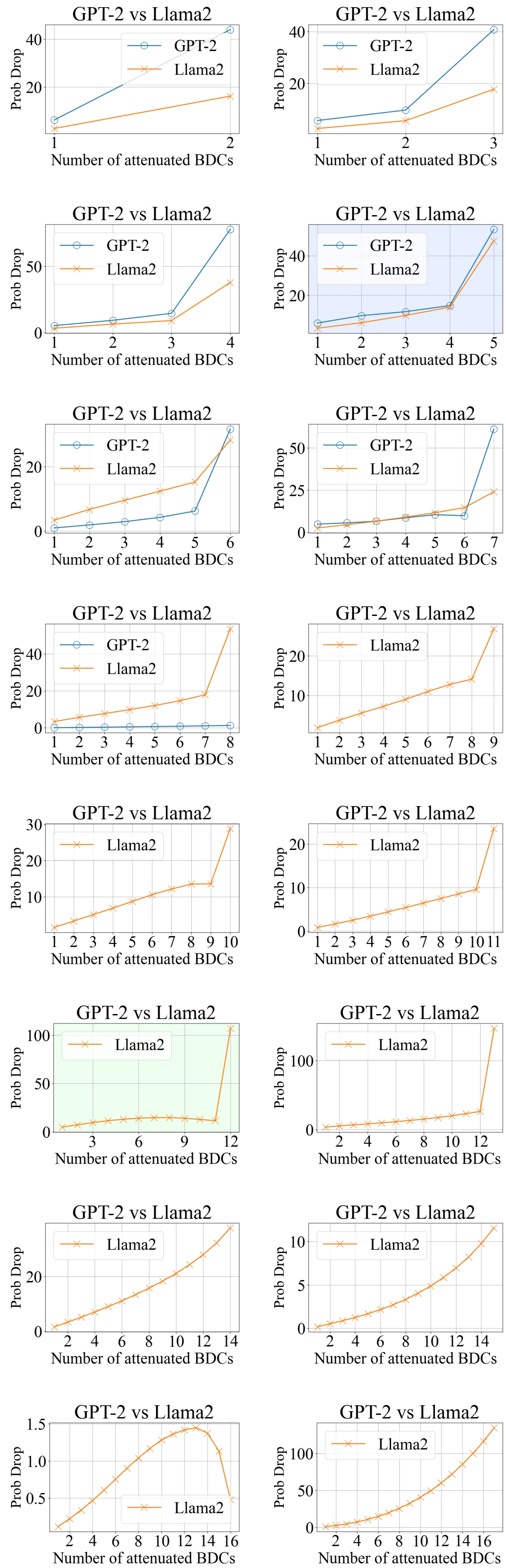

Figure 9: The graph of $\Delta Prob$ relative to the number of attenuated BDCs obtained using the NTC method.

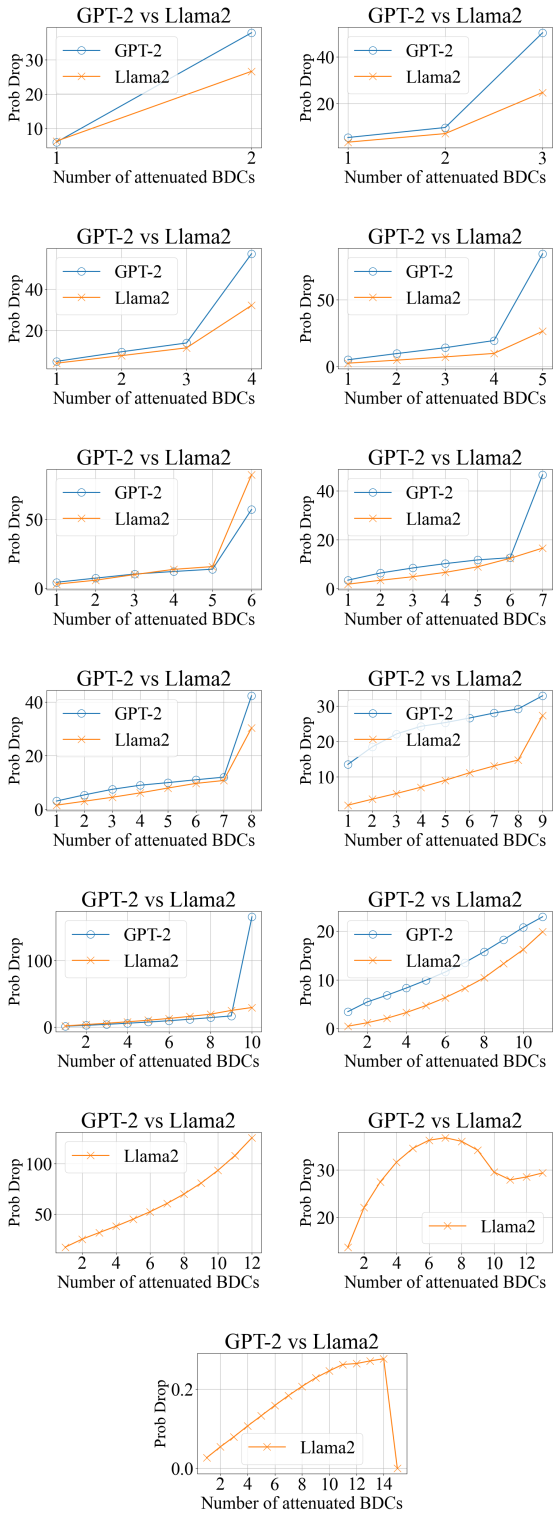

Figure 10: The graph of $\Delta Prob$ relative to the number of attenuated BDCs obtained using the DBSCAN method.

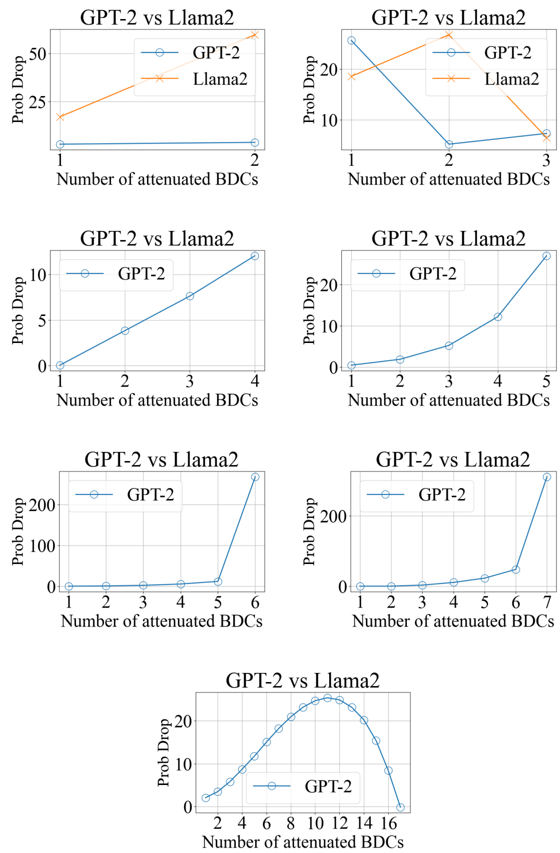

Figure 11: The graph of $\Delta Prob$ relative to the number of attenuated BDCs obtained using the Hierarchical method.

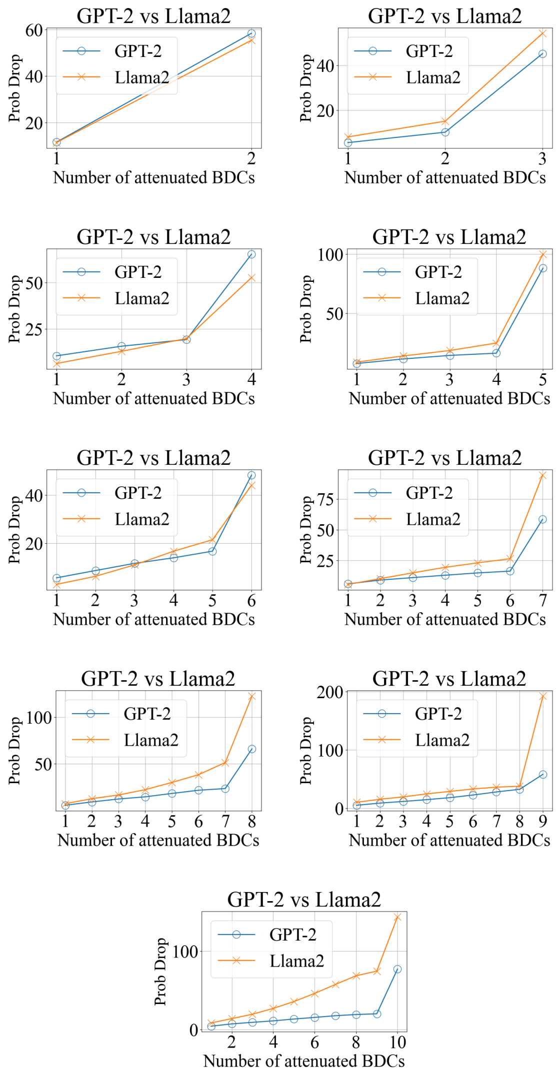

Figure 12: The graph of $\Delta Prob$ relative to the number of attenuated BDCs obtained using the K-Means method (Part 1).

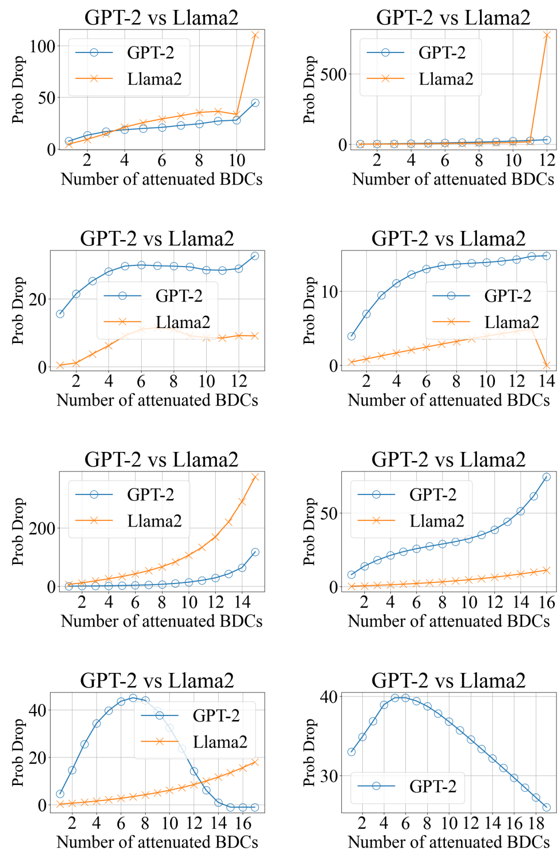

Figure 13: The graph of $\Delta Prob$ relative to the number of attenuated BDCs obtained using the K-Means method (Part 2).

Method	GPT-2						
	1	2	3	4	5	6	7
DBSCAN	—	12 → 26	17.7 → 38.8	23.4 → 33.8	32.9 → 53.4	24.8 → 49.8	26.4 → 46.6
Hierarchical	—	2.9 → 3.8	15.5 → 7.3	3.7 → 12	4.3 → 27	25 → 92	—
K-means	—	27.4 → 38.3	17.8 → 29.5	27.8 → 44.6	32.4 → 58.7	28.9 → 30.5	21.8 → 30.1
AMIG	—	-0.79 → 0.89	—	—	—	—	—
NTC (Ours)	—	**7.9 → 53**	**8.6 → 44**	**12 → 92**	**11 → 53**	**3.1 → 32**	**7.8 → 61**

Method	GPT-2						
	8	9	10	11	12	13	14
DBSCAN	9.1 → 12.5	30.4 → 32.9	39.8 → 19.3	11 → 22.9	—	—	—
Hierarchical	—	—	—	—	—	—	—
K-means	23.3 → 33.6	25.1 → 58.4	18.5 → 47.3	31 → 45	10.3 → 32.1	40.6 → 32.7	13 → 14.8
AMIG	—	—	—	—	—	—	—
NTC (Ours)	**0.6 → 1.3**	—	—	—	—	—	—

Method	GPT-2					
	15	16	17	18	19	Overall
DBSCAN	—	—	—	—	—	23.90 → 34.72
Hierarchical	—	—	20.8 → -0.13	—	—	20.77 → 25.05
K-means	48.8 → 118.1	29.2 → 74.7	62.7 → -1	—	37 → 26	34.42 → 38.86
AMIG	—	—	—	—	—	-0.79 → 0.89
NTC (Ours)	—	—	—	—	—	**9.32 → 55.60**

Method	Llama2					
	1	2	3	4	5	6
DBSCAN	—	7.1 → 15.4	8.7 → 15.2	11.2 → 21.0	8.9 → 14.6	17.6 → 34.2
Hierarchical	—	27.6 → 44.3	22.3 → 6.5	—	—	—
K-means	—	2.8 → 16.1	19.2 → 39.1	22.2 → 45.4	25.8 → 41.4	14.2 → 34.2
AMIG	—	-0.99 → 1.99	—	—	—	—
NTC (Ours)	—	**2.8 → 15.6**	**4.3 → 19.1**	**6.4 → 37.4**	**8.4 → 49.9**	**9.8 → 29.6**

Method	Llama2					
	7	8	9	10	11	12
DBSCAN	7.7 → 18.0	7.7 → 16.3	12.4 → 28.6	12.2 → 32.1	7.2 → 25.3	53.6 → 125.6
Hierarchical	—	—	—	—	—	—
K-means	28.6 → 50.8	37.9 → 97.1	29.4 → 74.9	47.9 → 87.8	50.7 → 110.5	13.5 → 23.2
AMIG	—	—	—	—	—	—
NTC (Ours)	**9.1 → 27.6**	**7.8 → 50.1**	**7.6 → 21.5**	**9.1 → 31.5**	**4.6 → 25.7**	**15.2 → 125.5**

Method	Llama2					
	13	14	15	16	17	Overall
DBSCAN	92.5 → 29.4	—	—	0.2 → 0	—	22.26 → 18.24
Hierarchical	9.7 → 9.1	36.3 → 0	—	—	—	27.50 → 44.04
K-means	9.7 → 9.1	36.3 → 0	68.8 → 375.0	3.3 → 11.1	4.9 → 17.9	20.08 → 39.24
AMIG	—	—	—	—	—	-0.99 → 1.99
NTC (Ours)	**15.8 → 183.7**	**13.6 → 37.7**	**3.1 → 17.2**	**1.0 → 0.5**	**31.2 → 134.9**	**14.11 → 28.78**

Table 6: The full results of Comparison of the degeneracy properties of DKN using different methods. "—" denotes methods that do not yield $\mathcal{D}$ of a specific length.